Сравнение способов первичного описания кода программы в задаче поиска похожих последовательностей кода

А.С. Юмаганов[a], В.В. Мясников[a]

[a] *Самарский национальный исследовательский университет имени академика С.П. Королева, 443086, Московское шоссе, 34, Самара, Россия*

Аннотация

В работе рассматривается задача поиска похожих последовательностей кода в исполняемых бинарных файлах. В рамках предложенного ранее оригинального метода поиска, кратко излагаемого в настоящей работе, исследуется влияние на его эффективность нескольких способов первичного описания кода программы. Представлены результаты экспериментальных исследований эффективности поиска для каждого из них, включающие сравнения по качественным показателям и затрачиваемым вычислительным ресурсам. Проведен анализ полученных результатов, представлено сравнение с известными решениями, демонстрирующее преимущества предложенных решений.

Ключевые слова: поиск; последовательность кода; команды процессора; оценка плотности распределения; наибольшая общая подпоследовательность

1. Введение

При разработке нового программного обеспечения программисты часто используют библиотеки/программные модули/отдельные функции, которые были реализованы ранее другими разработчиками [1]. Такая методология проектирования ПО называется «повторное использования кода» (code reuse). Она позволяет разработчикам нового ПО экономить большое количество ресурсов на разработке тех программных модулей, требуемый функционал которых уже был ранее реализован. Однако, такой подход имеет серьезные недостатки. Одним из таких недостатков является высокий риск переноса ошибок и уязвимостей из заимствованного модуля. Другой серьезной проблемой может стать возможность нарушения лицензионного соглашения на использование ПО. Решение задачи поиска похожих последовательностей кода может применяться не только для детектирования вредоносных программ (как было отмечено в работе [2]), но и для поиска ошибок и уязвимостей в программах и выявления нарушений лицензионного соглашения разработчиками.

Настоящая работа является продолжением прежних исследований авторов ([2]), посвященных разработке метода решения задачи поиска похожих последовательностей кода в исполняемых файлах. В данной работе подробно рассматриваются способы построения первичного описания кода исследуемой функции, сравнивается их эффективность.

Работа построена следующим образом. Во втором разделе дано краткое описание представленного ранее метода поиска похожих функций. Материалы данного раздела в развернутой форме можно найти в работе авторов [2]. В третьем разделе подробно описаны способы получения первичного описания функций исполняемого файла, на основе которого затем формируется окончательное описание функций. В четвертом разделе кратко описан способ оценки эффективности метода поиска похожих функций и приведены результаты экспериментальных исследований. В заключении приводятся выводы и список использованной литературы.

2. Основные понятия и принцип работы

Описанный в нашей предыдущей работе [2] метод поиска похожих функций использует следующие базовые понятия:

- текущая библиотека – набор функций исследуемого исполняемого файла;
- архивные данные – набор известных функций и их описаний через библиотеку базисных функций;
- базисная библиотека – вспомогательный набор функций, применяемый для сравнения функций архивных данных и текущей библиотеки.

Задача, решаемая этим методом, формулируется следующим образом: для заданной функции текущей библиотеки найти наиболее похожую функцию из архивных данных. В рамках данной работы рассматривается несколько вариантов выбора меры сходства, основанных на расположении функциональных групп команд в теле функции. Подробное описание каждого из них представлено в третьем разделе.

Разработанный ранее метод поиска похожих функций включает в себя несколько этапов. На первом этапе происходит представление функций архивных данных через библиотеку базисных функций и сохранение полученных описаний в архивной базе данных (БД). На втором этапе аналогичным образом формируется БД текущей библиотеки. На заключительном третьем этапе осуществляется непосредственно поиск похожих функций, в результате которого для заданной функции текущей библиотеки формируется упорядоченный по критерию евклидовой близости список функций архивных данных.

Предварительно все команды процессора разбиваются на K=47 функциональных групп [3], каждая из которых содержит команды, предназначенные для выполнения операций одного вида. В результате чего на базисную библиотеку накладывается следующее требование: каждая из K функциональных групп команд должна быть представлена как минимум в одной из функций библиотеки базисных функций.

Рассмотрим подробнее процесс представления функций текущей библиотеки и архивных данных через библиотеку базисных функций.

3. Построение первичного и окончательного описания функций

Для построения первичного описания рассматриваемой функции через библиотеку базисных функций необходимо получить значения некоторой меры схожести с каждой из базисных функций. Это может быть сделано многими различными способами. Исследования, проводимые в рамках настоящей работы, используют один из следующих:

1) пространственное распределение команд для каждой функциональной группы в дифференциальной форме (гистограмма);

2) пространственное распределение команд для каждой функциональной группы в интегральной форме;

3) ядерную оценку пространственного распределения команд для каждой функциональной группы (способ был использован в предыдущей работе авторов [2]);

4) длину наибольшей общей подпоследовательности списка номеров групп команд;

Рассмотрим подробно процесс получения каждого из представленных выше объектов сравнения и процесс формирования первичного и окончательного описания исследуемой функции.

3.1. Пространственное распределение команд для каждой функциональной группы в дифференциальной (гистограмма) и интегральной форме

Для каждой группы команд исследуемой функции строится пространственное распределения команд, входящих в эту группу, по длине функции следующим образом.

Пусть $n_0^k,..., n_{N_k-1}^k$ – абсолютные смещения относительно начала функции команд группы k внутри рассматриваемой функции, N_k – общее количество команд этой группы в данной функции, N – длина функции. Определим пространственное распределение k-го типа команд в дифференциальной форме как абсолютную частоту попадания команд этого типа в некоторый (относительный) приведенный i-ый интервал $(I = 100)$:

$$\tilde{f}_i^k = \sum_{j=0}^{N_k-1} I\left(\frac{n_j^k}{N}\cdot 100 \in (i-1, i]\right), \quad i = \overline{1, I}\ , \tag{1}$$

здесь I(.) - индикатор события, принимающий значения "0" или "1" в зависимости от истинности соответствующего аргумента.

Для получения пространственного распределения команд в интегральной форме воспользуемся следующей формулой:

$$\hat{f}_i^k = \frac{\sum_{y=0}^{i} \tilde{f}_y^k}{\sum_{j=0}^{I} \tilde{f}_j^k}, i = \overline{1, I} \tag{2}$$

Выбрав одну из форм представления пространственного распределения команд в теле функции ((1) или (2)), получаем K векторов приведенного ниже вида, каждый из которых характеризует пространственное положение команд k-ой группы в теле функции:

$$\bar{a}_k = \left(f_1^k, f_2^k, \ldots, f_I^k\right)^T, k = \overline{0, K-1}. \tag{3}$$

Тогда матрица описания, характеризующая положение команд каждой из K групп внутри функции, примет следующий вид:

$$A = \left(\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_{K-1}\right). \tag{4}$$

Аналогичную матрицу, именуемую далее C, можно построить для каждой функции базисной библиотеки. Мера схожести двух функций, задаваемых матрицами A и C, назовем следующую величину:

$$\mu(A,C)=\sum_{k=0}^{K-1}\alpha_k\mu_{\cos}(\bar{a}_k,\bar{c}_k)\ , \qquad (5)$$

где $\mu_{\cos}(\bar{a},\bar{c})$ – косинусная метрика, а величины $\alpha_k \geq 0$, удовлетворяющие условию

$$\sum_{k=0}^{K-1}\alpha_k = 1$$

– суть параметры метода. При совпадении двух функций мера схожести (5) принимает значение «1», в противном случае – «0».

Пусть далее J – число функций в базисной библиотеке, каждая из которых имеет описание в виде матрицы Cj. Тогда, сравнивая исследуемую функцию текущей библиотеки с каждой из функций базисной библиотеки, получим следующий вектор первичного описания этой функции:

$$\bar{x}_A=\left(\mu(A,C_0),\mu(A,C_1),\ldots,\mu(A,C_{J-1})\right)^T . \qquad (6)$$

Так как количество функций в библиотеке базисных функций J >> 1, воспользуемся методом снижения размерности – методом главных компонент. В результате получим вектор z размерности $I<J$, который выступает в качестве окончательного представления исследуемой функции через набор функций базисной библиотеки:

$$\bar{z}=Y\bar{x}\,, \qquad (7)$$

где Y – матрица перехода к новой размерности, составленная из I собственных векторов.

3.2. Ядерная оценка пространственного распределения команд для каждой функциональной группы

Для устранения влияния на результат поиска малых смещений команд в коде при использовании формулы (1) воспользуемся ядерной оценкой плотности распределения (kernel density estimation) [4]. Тогда оценка распределения команд в коде имеет вид:

$$f_i^k=\sum_{j=1}^{N}\tilde{f}_j^k\,\frac{1}{h}K\left(\frac{i-j}{h}\right),\quad i=\overline{0,I}\ ,$$

где $K(r)$ – функция ядра, h – ширина окна. В качестве функции ядра будем использовать ядро Гаусса:

$$K(r)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}r^2\right).$$

Как и в предыдущем разделе воспользуемся формулами (3), (4), (5) для получения представления рассматриваемой функции через библиотеку базисных функций в виде вектора (6), которое затем сократим с помощью метода главных компонент (7). В результате получим вектор окончательного описания функции.

3.3. Длина наибольшей общей подпоследовательности (НОП)

Сформируем упорядоченную по смещениям относительно начала функции последовательность номеров групп соответствующих команд для рассматриваемой функции. Аналогичным образом получим последовательность номеров групп для функций базисной библиотеки.

Рассмотрим последовательность чисел $a_1,a_2,...,a_n$. Если из данной последовательности чисел удалить часть элементов, то получим новую последовательность, которую называют подпоследовательностью исходной последовательности. Рассмотрим еще одну последовательность чисел $b_1,b_2,...,b_n$. Пусть требуется найти длину самой длинной подпоследовательности последовательности $\{a_n\}$, которая одновременно является и подпоследовательностью последовательности $\{b_n\}$. Такую последовательность называют наибольшей общей подпоследовательностью (НОП) двух последовательностей.

Пусть последовательность $\{a_n\}$ номеров групп первой функции имеет длину N_1, а последовательность $\{b_n\}$ номеров групп второй функции – N_2. Мерой схожести двух функций, задаваемых последовательностями *a,b* назовем следующую величину:

$$\mu(\{a_n\},\{b_n\})=\frac{2*ДНОП(\{a_n\},\{b_n\})}{N_1+N_2}\ ,$$

где ДНОП – длина НОП. При полном совпадении функций мера схожести принимает значение "1", при полном несоответствии – "0".

Для решения задачи нахождения ДНОП воспользуемся алгоритмом Хиршберга [5]. Работа алгоритма сводится к поэтапному заполнению матрицы L, строки которой представляют собой элементы последовательности $\{a_n\}$, а столбцы – элементы последовательности $\{b_n\}$.

Матрица L заполняется следующим образом:

$$L(i,j)=\begin{cases}0, i=0 \vee j=0\\ L(i-1,j-1)+1, a(i)=b(j)\\ \max(L(i-1,j),L(i,j-1)), a(i)\neq b(j)\end{cases}.$$

Таким образом, в ячейке *(i,j)* матрицы L хранится длина НОП для последовательностей $\{a_n\}_{n=1}^{i}$ и $\{b_n\}_{n=1}^{j}$. В результате работы алгоритма получим матрицу L, элемент $L(N_1, N_2)$ которой содержит искомую длину НОП двух последовательностей.

Пусть J – число функций в библиотеке базисных функций. Сравнивая исследуемую функцию с каждой из функций библиотеки базисных функций, получим следующий вектор первичного описания данной функции:

$$\bar{x}_{\{a_n\}}=\left(\mu\left(\{a_n\},\{b^0{}_n\}\right),\ldots,\mu\left(\{a_n\},\{b^{J-1}{}_n\}\right)\right)^T. \quad (8)$$

Затем воспользуемся формулой (7) для получения вектора окончательного представления рассматриваемой функции.

4. Результаты экспериментов

Для проведения экспериментальных исследований использовался стандартный ПК (Intel Core i5- 3450 3.1 ГГц, 16 Гб ОЗУ).

Для оценки эффективности разработанного метода поиска схожих функций воспользуемся в качестве текущей библиотеки некоторой динамической библиотекой (например, libtiff), а в качестве архивной - такой же динамической библиотекой только другой версии. Установим минимальный размер функции равный 16 байтам, так как рассматриваемые библиотеки содержат большое количество «маленьких» функций (1-3 инструкции), содержащих одинаковые команды.

Положим, что при модификации кода библиотеки (при переходе от одной версии к другой) имена функций не менялись, и имена функций архивных данных не повторяются. Для проверяемой функции текущей библиотеки получим список похожих функций архивных данных $\{F_l\}_{l=1,\ldots,L}$, отсортированный по уменьшению схожести. Алгоритм получения этого списка подробно описан в нашей предыдущей работе [2]. При указанных допущениях оценить качество распознавания можно поставив в соответствие этому списку функций бинарную последовательность $\beta=(\beta_1,\beta_2,\ldots,\beta_L)$, такую, что при наличии на l-ой позиции функции с тем же именем, что и у проверяемой, $\beta_l=1$, иначе – $\beta_l=0$. Воспользуемся следующими критериями оценки качества информационного поиска [6][7]:

- *Точность для k-ой позиции списка :* $P_k=\dfrac{\sum_{l=1}^{k}\beta_l}{k}$
- *Полнота для k-ой позиции списка :* $R_k=\dfrac{\sum_{l=1}^{k}\beta_l}{K}$
- *Средняя точность для списка:* $AveP=\sum_{k=1}^{L}P_k(R_k-R_{k-1}), R_0=0$.

Тогда средняя точность для всех функций входящих в состав текущей библиотеки вычисляется по формуле:

$$P=\frac{1}{S}\sum_{s=0}^{S-1}AveP_s, \quad (9)$$

где S – количество функций в текущей библиотеке.

Пусть архивные данные представлены динамической библиотекой libtiff 4.0.3 [8], а текущая библиотека представлена одной из следующих версий библиотеки libtiff: libtiff 3.9.7, libtiff 4.0.4, libtiff 4.0.5, libtiff 4.0.6. В каждой из данных библиотек содержится около 1200 функций, размер которых превышает 16 байт.

Библиотека базисных функций содержит J=50 функций, выбранных вручную из различных исполняемых файлов. Для каждой из K=47 функциональных групп команд существует как минимум одна функция базисной библиотеки, содержащая команды этой группы.

Вектор окончательного описания функции через библиотеку базисных функций имеет размер I=5.

Основным параметром представленного метода поиска похожих функций является выбор сравниваемой меры (объекта сравнения). Как отмечалось ранее, в качестве такой меры могут выступать:

- пространственное распределение команд в теле исследуемой функции (в дифференциальной или в интегральной форме);
- ядерная оценка пространственного распределения команд в теле исследуемой функции;
- длина наибольшей общей подпоследовательности групп команд в теле функции.

Сравним значения средней точности поиска (9) для четырех версий библиотеки libtiff при использовании рассмотренных выше объектов сравнения. В таблице 1 представлены полученные результаты.

Таблица 1. Сравнение точности поиска для различных объектов сравнения

Текущая библиотека	Пространственное распределение команд в теле функции в дифференциальной форме	Пространственное распределение команд в теле функции в интегральной форме	Ядерная оценка пространственного распределения команд в теле функции (h = 3)	Длина НОП групп команд
libtiff 3.9.7	0,7087	0,7854	0,7821	0,7276
libtiff 4.0.4	0,8406	0,8515	0,8496	0,8126
libtiff 4.0.5	0,8377	0,8499	0,8479	0,8117
libtiff 4.0.6	0,8369	0,8428	0,8391	0,8052

Наименьшая средняя точность поиска в большинстве случаев получается при использовании в качестве объекта сравнения длины наибольшей общей подпоследовательности (ДНОП) групп команд. Это связано с тем, что в случае сравнения на основе пространственного распределения команд в теле функции учитывается не только последовательность команд, но и размеры инструкций и их операндов, так как значение длины функции в формуле (1) зависит от этих значений. В случае ДНОП учитывается только порядок инструкций в теле рассматриваемой функции, что снижает точность поиска похожих функций. Лучший результат был получен при использовании пространственного распределения команд в интегральной форме в качестве объекта сравнения.

Рассмотрим среднее время, затрачиваемое на построение архивной и текущей баз данных, для каждого из рассмотренных объектов сравнения. В таблице 2 представлены полученные результаты. Минимальное время построения баз данных получается при использовании в качестве объекта сравнения пространственного распределения команд в теле функции в дифференциальной форме. При использовании в качестве объекта сравнения длины НОП групп команд время, затрачиваемое на построение БД, увеличивается почти в 5 раз по сравнению с лучшим результатом. Другие два объекта сравнения (пространственное распределение команд в интегральной форме и ядерная оценка пространственного распределения команд) показали результат, уступающий лидеру примерно в 3 раза.

Таблица. 2. Сравнение времени построения БД

Объект сравнения	Среднее время построения БД, с
Пространственное распределение команд в теле функции в дифференциальной форме	342
Пространственное распределение команд в теле функции в интегральной форме	1023
Ядерная оценка пространственного распределения команд в теле функции	1135
Длина НОП групп команд	1502

Результаты экспериментов показали, что лучшим объектом сравнения по критерию максимальной средней точности поиска является пространственное распределение команд в теле функции в интегральной форме, по критерию времени – пространственное распределение команд в теле функции в дифференциальной форме.

Сравним эффективность представленного в данной работе метода поиска похожих функций (объект сравнения – пространственное распределение команд в теле функции в интегральной форме) и другого известного метода поиска похожих функций – алгоритма распознавания библиотечных функций IDA FLIRT (Fast Library Identification and Recognition Technology)[9]. Полученные результаты представлены в таблице 3.

Анализ данных таблицы 3, показывает, что предложенный авторами оригинальный метод поиска похожих функций превосходит названный выше алгоритм по показателю средней точности поиска P.

Таблица 3. Сравнение методов поиска похожих функций

Текущая библиотека	Средняя точность поиска P похожих функций, используя разработанный метод	Средняя точность поиска P похожих функций, используя алгоритм IDA FLIRT
libtiff 3.9.7	0,7854	0,5035
libtiff 4.0.4	0,8515	0,8006
libtiff 4.0.5	0,8499	0,7912
libtiff 4.0.6	0,8428	0,7656

5. Заключение

В рамках решения задачи поиска похожих последовательностей кода в исполняемых бинарных файлах и предложенного ранее оригинального метода поиска в работе исследуются несколько способов построения первичного описания кода. Приводятся результаты экспериментальных исследований, на основе которых делаются выводы об их сравнительной эффективности по показателям качества и затрачиваемым вычислительным ресурсам. Продемонстрировано превосходство оригинального метода над известным алгоритмом IDA FLIRT поиска похожих последовательностей кода в исполняемых бинарных файлах. Дальнейшие исследования будут направлены на дальнейшее повышение эффективности предложенного оригинального метода.

Благодарности

Работа выполнена при частичной финансовой поддержке гранта РФФИ, проект 15-07-01164-а.

Литература

[1] Zaimi, A. An Empirical Study on the Reuse of Third-Party Libraries in Open-Source Software Development / A. Zaimi , A. Ampatzoglou , N. Triantafyllidou, A. Chatzigeorgiou , A. Mavridis , T. Chaikalis , I. Deligiannis , P. Sfetsos , I. Stamelos // Proceedings of the 7th Balkan Conference on Informatics Conference. – 2015. DOI:10.1145/2801081.2801087.

[2] Yumaganov, A.S. Similarity search over program code sequences using featureless pattern recognition techniques / A.S. Yumaganov, V.V. Myasnikov // CEUR Workshop Proceedings. – 2016. – Vol. 1638. – P. 437-443. DOI: 10.18287/1613-0073-2016-1638-437-443.

[3] x86 Assembly Language Reference Manual [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/cd/E19253-01/817-5477/817-5477.pdf (14.01.2017)

[4] Фукунага, К. Введение в статистическую теорию распознавания образов [Текст]: Пер. с англ./ К. Фукунага – М.: Наука. Главная редакция физико-математической литературы, 1979. – 171-173 с.

[5] Hirschberg, D.S. A linear space algorithm for computing maximal common subsequences / D.S. Hirschberg // Communications of the ACM. – 1975. – Vol. 18. No. 6. – P. 341-343.

[6] Buckland, M. K. The relationship between recall and precision/ M.K. Buckland, F.C. Gey // JASIS. – 1994. – Vol. 45. No. 1. – P. 12-19.

[7] Powers, D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. // Journal of Machine Learning Technologies. – 2011. – Vol. 2. No. 1. – P. 37-63.

[8] TIFF Library and Utilities [Электронный ресурс]. – Режим доступа: http://www.libtiff.org/ (14.01.2017).

[9] IDA F.L.I.R.T Technology: In-Depth [Электронный ресурс]. – Режим доступа: https://www.hexrays.com/products/ida/tech/flirt/in_depth.shtml (14.01.2017).